VBmix: a R package for Variational-Bayes mixture learning

Pierrick Bruneau

VBmix: a R package for Variational-Bayes mixture learning

Pierrick Bruneau

Abstract

This paper documents VBmix, a R package available on CRAN (`http://cran.r-project.org/web/packages/VBmix/index.html`).

Introduction

VBmix regroups several learning algorithms, and associated tools (data preprocessing, plotting).

The following functions are the entry points to using this piece of software:

- **classicEM**: the classic EM algorithm for learning Gaussian mixtures [3].
- **varbayes**: the variational Bayesian algorithm for learning Gaussian mixtures. The algorithm learns the optimal number of Gaussian components, along with the model parameters [3].
- **vbcomp**: the variational Bayesian algorithm for aggregating Gaussian mixtures to the optimal number of components, building upon the principles laid out in varbayes [4].
- **vbconstr**: variant of the variational Bayesian aggregation algorithm, that allows specifying constraints for aggregation [4].
- **mppca**: the variational Bayesian algorithm for learning probabilistic PCA mixtures. The algorithm learns the optimal number of components, and their intrinsic dimensionality [14, 2, 5].
- **mmppca**: the variational Bayesian algorithm for aggregating probabilistic PCA mixtures, inspired by mppca [5].
- **gmmkmsock**: k-means algorithm operating on objects represented by Gaussian mixtures [6].

Installation

To install VBmix, simply download the latest package source at the URL indicated in the abstract. Then, assuming 0.2.9 as the current version, type the following:

```
sudo R CMD INSTALL VBmix_0.2.9.tar.gz
```

Prior to this installation, ensure that the pre-requisites indicated on the CRAN page (especially having Qt, GSL and fftw3 installed) are met. The rest of the paper gives the comprehensive manual of the functions contained in the package, in alphabetic order.

`appendToGmm` *appendToGmm*

Description

concatenates mod2 to mod1.

Usage

```
appendToGmm(mod1, mod2)
```

Arguments

`mod1`	GMM to which mod2 is appended.
`mod2`	GMM appended to mod1.

Value

GMM with concatenated models, with a set accordingly.

Examples

```
temp <- appendToGmm(gmmpen[[1]], gmmpen[[2]])
```

`appendToList` *appendToList*

Description

appends 1 list object to another.

Usage

```
appendToList(lst, obj, appendList = FALSE)
```

Arguments

`lst`	list object to which we append an object.
`obj`	object to append.
`appendList`	if TRUE, obj should be a list object, which elements are appended. if FALSE, obj is simply added to lst.

Value

list object with obj appended to lst.

See Also

appendToGmm appendToMppca

Examples

```
temp <- list()
temp <- appendToList(temp, pcapen[[1]]$mumean, appendList=TRUE)
temp <- appendToList(temp, pcapen[[2]]$mumean, appendList=TRUE)
```

---

appendToMppca *appendToMppca*

Description

appends mppca2 to mppca1.

Usage

```
appendToMppca(mppca1, mppca2)
```

Arguments

mppca1	MPPCA model to be appended to.
mppca2	MPPCA to append to mod1.

Value

appended models.

See Also

appendToGmm appendToList

Examples

```
temp <- appendToMppca(pcapen[[1]], pcapen[[2]])
```

`binnedEntropy` *binnedEntropy*

Description

uses bins to approximate the empirical entropy of a variable.

Usage

```
binnedEntropy(v, nbins = 100)
```

Arguments

`v`	a numeric vector.
`nbins`	number of bins used to estimate the entropy.

Value

entropy value.

Examples

```
temp <- binnedEntropy(irisdata[,1])
```

`buildFrame` *buildFrame*

Description

builds a data frame from a matrix of elements and a vector of numeric labels.

Usage

```
buildFrame(datamatrix, labels, dims = 1:2)
```

Arguments

`datamatrix`	matrix of row-elements.
`labels`	vector of numeric labels.
`dims`	subset of variables extracted from datamatrix.

Value

built data frame.

Examples

```
irisdata[c(1,7,35,56,131),]
# returns:
#         Sepal.Length Sepal.Width Petal.Length Petal.Width
#[1,]          5.1         3.5          1.4         0.2
#[2,]          4.6         3.4          1.4         0.3
#[3,]          4.9         3.1          1.5         0.2
#[4,]          5.7         2.8          4.5         1.3
#[5,]          7.4         2.8          6.1         1.9
irislabels[c(1,7,35,56,131)]
# returns:
#[1] 1 1 1 2 3
temp <- buildFrame(irisdata, irislabels, dims=1:4)
```

`circlegen` *circlegen*

Description

generate data elements along a 2D circle with additional noise.

Usage

```
circlegen(npts = 200, radius = 10, noise = 1)
```

Arguments

`npts`	number of elements to generate.
`radius`	radius of the circle.
`noise`	determines the width of the circle stroke.

Value

matrix of sampled row-elements.

Examples

```
temp <- circlegen()
```

classicEM *classicEM*

Description

estimates a GMM on data using EM algorithm. A lower bound is calculated and monitored at each iteration.

Usage

```
classicEM(data, k, thres = 0.1, maxit = NULL)
```

Arguments

data	matrix of row-elements.
k	maximal number of components in the GMM. In case of degeneracies, the final model size may be less than 0.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.

Value

estimated GMM with at most k components, with labels containing associated labels for data in addition.

References

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006

See Also

newGmm varbayes

Examples

```
temp <- classicEM(irisdata, 4)
```

constrClassif *constrClassif*

Description

performs task analogous to mixKnn (i.e. leave-one-out classification), but uses synthetic representatives to infer labels, instead of k-NN. Each representative is obtained by concatenating all GMM (i.e. elements) of a specific label value, and applying vbconstr on this redundant mixture.

Usage

```
constrClassif(data, labels, KLparam = 500, rho = new.env())
```

Arguments

data	list of GMM.
labels	vector of numeric labels associated to data.
KLparam	number of samples for jsmc.
rho	R environment object. Used to issue R commands within the C routine.

Value

classification error ratio in [0,1].

See Also

mixKnn vbconstr

Examples

```
temp1 <- sample(1:200, 150)
temp2 <- list()
for(i in temp1) temp2 <- appendToList(temp2, imgmods[[i]])
temp3 <- imglabels[temp1]
# de-activated because this process is very long...
#temp4 <- constrClassif(temp2, temp3)
```

covgen *covgen*

Description

generates random definite positive matrices (i.e. valid covariance matrices).

Usage

```
covgen(d = 2, bounds = c(1, 5))
```

Arguments

d	rank of the square matrix to be returned.
bounds	minima and maximal values for diagonal values.

Value

random definite positive matrix

Note

Matrix cells are sampled with an heuristic not guaranteed to lead to definite positiveness: this characteristic is only controlled before function return. If positive definite after control, the matrix is returned. If not, an error message is issued.

See Also

randomGmm

Examples

```
temp <- covgen()
```

dat1sample *dat1sample*

Description

generates data elements according to SYN1 process (sample from a 2D GMM, linearly transformed with additive noise, see reference).

Usage

```
dat1sample(nelts, gmm, noise, transform, oldbounds = NULL, newbounds = NULL)
```

Arguments

nelts	number of elements to generate.
gmm	2D GMM to be sampled from.
noise	additive noise magnitude.
transform	matrix defining linear transform.
oldbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as oldspan.
newbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as newspan.

Value

matrix of sampled row-elements

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

dat2sample dat3sample

Examples

```
temp <- dat1sample(500, randomGmm(), 1, generate2Dtransform())
```

---

dat2sample *dat2sample*

Description

generates data elements according to SYN2 process (sample along a semi-sphere with additive noise, see reference).

Usage

```
dat2sample(nelts, radius, noise, oldbounds = NULL, newbounds = NULL)
```

Arguments

nelts	number of elements to generate.
radius	radius of the sphere to sample from.
noise	additive noise magnitude.
oldbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as oldspan.
newbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as newspan.

Value

matrix of sampled row-elements.

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

dat1sample dat3sample

Examples

```
temp <- dat2sample(500, 10, 1)
```

---

dat3sample *dat3sample*

Description

generates data elements according to SYN3 process (sample along a 2D circle with additive noise, and linearly transform to higher dimensional space with further noise addition, see reference).

Usage

```
dat3sample(nelts, radius, noise, transform, oldbounds = NULL, newbounds = NULL)
```

Arguments

nelts	number of elements to generate.
radius	radius of the sphere to sample from.
noise	additive noise magnitude.
transform	matrix defining linear transform.

oldbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as oldspan.
newbounds	optional argument for sample rescaling. If not NULL, transmitted to setDomain as newspan.

Value

matrix of sampled row-elements.

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

dat1sample dat2sample

Examples

```
temp <- dat3sample(500, 10, 1, generate2Dtransform())
```

---

datagen *datagen*

Description

generates data from a random multivariate Gaussian, and adds redundant dimensions by random linear combinations with noise.

Usage

```
datagen(dreal = 2, deff = 6, npts = 200, noise = 0.1, genmean = rep(0, dreal), genspan = 6, iso = FALSE)
```

Arguments

dreal	dimensionality of the multivariate Gaussian.
deff	dimensionality of the returned sample.
npts	number of elements to be sampled.
noise	noise magnitude for the linear combination.
genmean	mean of the multivariate Gaussian.
genspan	maximal magnitude of the diagonal elements in the covariance matrix. Non-diagonal elements are sampled under constraints of positive-definiteness.
iso	sample from an isotropic multivariate Gaussian (i.e. diagonal covariance matrix).

Value

matrix of sampled row-elements.

Examples

```
temp <- datagen()
```

`dDirichlet` *dDirichlet*

Description

get density of a sample w.r.t Dirichlet distribution (3D only).

Usage

```
dDirichlet(alpha = 0.1, x1, x2)
```

Arguments

`alpha`	alpha parameter of the distribution (i.e. alpha repeated 3 times).
`x1`	1st dimension of the sample.
`x2`	2nd dimension of the sample.

Value

density value.

See Also

rDirichlet

Examples

```
temp <- dDirichlet(x1=0.4, x2=0.2)
# 3rd dimension is 1-x1-x2 = 0.2
```

displayGraph *displayGraph*

Description

displays a curve (vect, measure), and associated deviations. Typically used to present experimental results.

Usage

```
displayGraph(measure, dev, vect, xlab = "K", ylab = "measure", main = " ")
```

Arguments

measure	y-axis for the curve.
dev	deviations for the y-axis measures.
vect	x-axis for the curve.
xlab	label for x-axis.
ylab	label for y-axis.
main	main label for the plotting window.

Value

a new plotting window displaying the curve.

Examples

```
displayGraph(rnorm(10, mean=4, sd=3), rnorm(10, mean=0, sd=0.5), 1:10)
```

displayNnet *displayNnet*

Description

displays the colored decision regions of a neural network model. Data symbols are also optionally displayed. Data and model should be 2D.

Usage

```
displayNnet(nnet.model, datamatrix, datalabels, subset = NULL, displayPoints = TRUE, steps = 100, alpha
```

Arguments

nnet.model	a neural network model, as returned by nnet (nnet library)
datamatrix	a matrix of row-elements.
datalabels	matrix of binary indicator variables for labels (as used by nnet).
subset	vector of indexes of a data subset to be displayed. If NULL, all points are displayed.
displayPoints	if FALSE, only decision regions are displayed.
steps	influences the resolution of the decision regions. Low values will provoke aliasing, high values are slower to be displayed.
alpha	alpha blending parameter between decision regions and data symbols.
lwd	magnification factor for the stroke width used to plot symbols.

Value

a new plotting window displaying decision regions associated to the parametrized neural network.

See Also

nnet

Examples

```
temp <- class.ind(irislabels)
temp2 <- setDomain(irisdata[,1:2], 10)
temp3 <- nnet(temp2, temp, size=10)
displayNnet(temp3, temp2, temp)
```

---

displayScatter *displayScatter*

Description

general plotting function for data sets (matrix of row-elements), optionally associated to labels and a GMM. Labels influence the color and symbols of plotted data points. Gaussian envelopes of the components in the GMM are drawn. NB: data set and GMM arguments cannot be both NULL.

Usage

```
displayScatter(data = NULL, model = NULL, labels = NULL, datasizes = NULL, compcolors = NULL, complabels
```

Arguments

data	matrix of row-elements. If NULL, the GMM is plotted alone.
model	GMM object.
labels	vector of numeric labels. May alternatively be present as a member of model, labels.
datasizes	vector of integer magnification factors for data symbols. If length=1, same coefficient applies to all points.
compcolors	vector of integer color indexes. These indexes are internally associated to one color among a set of appropriately chosen ones. If length=1, all GMM components are colored the same way. If length=k, each component is associated to its own color index. This k-length vector may contain NA values: associated components will be white-colored.
complabels	character vector containing text strings to be printed over Gaussian envelopes.
compstrokes	this character vector may be used to specify non default strokes for envelopes.
space	this function prints a 2D scatterplot. If data and model have higher dimensionality, this argument specifies the axes to be printed.
xlim	bounds for the first variable. If NULL, will be inferred from available data.
ylim	bounds for the second variable. If NULL, will be inferred from available data.
main	main label for the plotting window.
xlab	label for the x-axis.
ylab	label for the y-axis.
smooth	if TRUE, display the response to a kernel density function, instead of symbols for data elements.
alphacol	alpha blending parameter when a component is non-white colored.
alphanocol	alpha blending parameter when a component is white colored.
cex.lab	magnification factor for all text in the plotting window.
lwd	width of the stroke used for data symbols.

Value

a new plotting window displaying the data set and associated model.

See Also

plotGmm

Examples

```
displayScatter(irisdata, NULL, irislabels)
```

displaySVM *displaySVM*

Description

displays the colored decision regions of a SVM model. Data symbols are also optionally displayed. Data and model should be 2D.

Usage

```
displaySVM(svm.model, dataframe, displayPoints = TRUE, subset = NULL, steps = 100, alpha = 0.4, lwd = 1)
```

Arguments

svm.model	a SVM model, as returned by svm (e1071 library)
dataframe	data.frame object, containing row-elements, and associated labels in the last variable.
displayPoints	if FALSE, only decision regions are displayed.
subset	vector of indexes of a data subset to be displayed. If NULL, all points are displayed.
steps	influences the resolution of the decision regions. Low values will provoke aliasing, high values are slower to be displayed.
alpha	alpha blending parameter between decision regions and data symbols.
lwd	magnification factor for the stroke width used to plot symbols.

Value

a new plotting window displaying SVM decision regions.

See Also

svm

Examples

```
# extract 2 first variables and build data.frame
temp <- buildFrame(irisdata, irislabels)
iris.model <- svm(labels ~ ., data=temp, cost=100, gamma=1)
displaySVM(iris.model, temp)
```

eigenMppca *eigenMppca*

Description

uses eigen decompositions to align factor matrices to principal bases (see references). NB: mppca and mmppca already perform this operation during their post-processing.

Usage

```
eigenMppca(mod)
```

Arguments

mod	MPPCA model which components have to be aligned.

Value

adjusted MPPCA.

References

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society - B Series*, 61(3):611–622, 1999

See Also

mppca newMppca

Examples

```
temp <- eigenMppca(pcapen[[2]])
```

extractSimpleModel *extractSimpleModel*

Description

extracts a GMM from a posterior variational distribution. Only relevant components (i.e. associated to a significant population) are extracted.

Usage

```
extractSimpleModel(model = model, labels = FALSE)
```

Arguments

model	variational posterior.
labels	boolean indicating wether to extract a label vector. If TRUE, model, a list object, should also contain a data attribute, used to build label vector.

Value

GMM object.

See Also

varbayes subVarbayes

Examples

```
temp <- varbayes(irisdata, 20)
temp2 <- extractSimpleModel(temp)
```

---

gaussianKL *gaussianKL*

Description

computes KL(N(0, Sigma_0) || N(0, Sigma_1)).

Usage

```
gaussianKL(N0, N1)
```

Arguments

N0	Sigma_0
N1	Sigma_1

Value

KL value.

See Also

klmc

Examples

```
temp <- gaussianKL(gmmpen[[1]]$cov[[1]], gmmpen[[1]]$cov[[2]])
```

generate2Dtransform *generate2Dtransform*

Description

generate a random matrix to transform a 2D signal to higher dimensional spaces.

Usage

```
generate2Dtransform(dims = 4)
```

Arguments

dims	dimensionality of the target space.

Value

a dims x 2 matrix defining the transform.

See Also

dat1sample dat3sample

Examples

```
temp <- generate2Dtransform()
```

generateSparsePoints *generateSparsePoints*

Description

generates a set of points pairwise-separated by a minimal distance. Is not guaranteed to converge: when maxit is reached, current points are returned.

Usage

```
generateSparsePoints(npoints, dim = 2, span = 10, mindist = 2, maxit = 20)
```

Arguments

npoints	number of points to generate (i.e. in a matrix with elements as rows.
dim	number of variables to generate.
span	[-span, span] is used as bounds to uniform sampling for all variables.
mindist	minimal distance that each element should have with all others. the "control" C routine is used to perform this verification. All points that do not respect this constraint are resampled.
maxit	maximal number of iterations before current elements are returned.

Value

matrix with well separated elements as its rows.

Examples

```
temp <- generateSparsePoints(10)
```

---

getBic *getBic*

Description

computes BIC criterion (see references) for a specific GMM and data set.

Usage

```
getBic(gmm, dat)
```

Arguments

gmm	GMM object.
dat	matrix of row-elements.

Value

BIC estimate.

References

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 61:461–464, 1978

See Also

getDataLikelihood varbayes

Examples

```
temp <- getBic(gmmpen[[1]], pendat)
```

getColor *getColor*

Description

associates a R color name (i.e. in the ouput of colors()) to each possible integer input index. Colors are chosen in a reduced, well differentiated, subset.

Usage

```
getColor(index)
```

Arguments

index	integer input index.

Value

color name.

Examples

```
getColor(3)
```

getCouple *getCouple*

Description

computes classification error function described in references, a.k.a couple error. In brief, evaluates how elements are gathered similarly, irrespectively of exact label values (adapted to clustering).

Usage

```
getCouple(vec1, vec2)
```

Arguments

vec1	vector of numeric labels.
vec2	vector of numeric labels.

Value

classification error in [0,1].

References

E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.*, 78:553–569, 1983

F. Picarougne, H. Azzag, G. Venturini, and C. Guinot. A new approach of data clustering using a flock of agents. *Evolutionary Computation*, 78(3):345–367, 2007

Examples

```
temp <- classicEM(irisdata, 4)
getCouple(temp$labels, irislabels)
#[1] 0.1524832
```

`getDataLikelihood` *getDataLikelihood*

Description

gets log-likelihoods associated to a matrix of row-elements.

Usage

```
getDataLikelihood(gmm, dat)
```

Arguments

`gmm`	GMM object.
`dat`	matrix of row-elements.

Value

numeric vector of log-likelihoods.

See Also

getBic gmmgen

Examples

```
temp <- getDataLikelihood(gmmpen[[3]], pendat)
```

getLabels *getLabels*

Description

gets numeric labels that associates a data set and a GMM.

Usage

```
getLabels(model, data)
```

Arguments

model	GMM.
data	matrix of row-elements.

Value

vector of numeric labels, that take values of the respective component indexes in the GMM.

See Also

newGmm

Examples

```
temp <- classicEM(irisdata, 4)
temp2 <- getLabels(temp, irisdata)
```

getQforComp *getQforComp*

Description

gets the rank associated with a properly aligned factor matrix.

Usage

```
getQforComp(loadings, tau = 1, verbose = FALSE, quick = FALSE)
```

Arguments

loadings	aligned factor matrix.
tau	diagonal noise used for KL computations.
verbose	if TRUE maximal info is displayed.
quick	if TRUE, column norm values are used instead of KL computations (less accurate but faster).

Value

rank associated with loadings.

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

newMppca mppca

Examples

```
temp <- getQforComp(pcapen[[1]]$wmean[[2]], quick=TRUE)
```

`getResp` *getResp*

Description

get posterior responsibilities of elements in a data set, according to a posterior MPPCA distribution.

Usage

```
getResp(data, model)
```

Arguments

`data`	matrix of row-elements.
`model`	posterior MPPCA.

Value

nxk matrix (with n the number of row-elements, and k the number of components in the MPPCA) of membership probabilities. (i.e. Z in references)

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

mppca

Examples

```
temp <- getResp(pendat, pcapen[[1]])
```

getVarbayesResp *getVarbayesResp*

Description

gets posterior responsibilities for a data set, according to the variational posterior of a GMM.

Usage

```
getVarbayesResp(data, model)
```

Arguments

data	matrix of row-elements.
model	variational posterior of a GMM

Value

responsibility matrix (Z in references) resulting from the parameters.

References

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006

See Also

getResp ZtoLabels

Examples

```
# get resp for only a subsample, as this operation is rather long.
temp <- getVarbayesResp(pendat[1:10,], vbpen[[2]])
```

gmmdensity *gmmdensity*

Description

get densities of a set of elements w.r.t a GMM.

Usage

```
gmmdensity(mod, data)
```

Arguments

mod	reference GMM.
data	matrix of row-elements.

Value

numeric vector containing densities.

See Also

gmmgen

Examples

```
temp <- gmmgen(gmmpen[[1]], 50)
temp2 <- gmmdensity(gmmpen[[1]], temp[[1]])
```

---

gmmgen *gmmgen*

Description

sample elements from a GMM.

Usage

```
gmmgen(mod, nitem)
```

Arguments

mod	GMM sampled from.
nitem	number of elements to be sampled.

Value

nitem x d matrix with elements as rows.

Examples

```
temp <- gmmgen(gmmpen[[1]], 50)
```

gmmkmsock *gmmkmsock*

Description

perform k-means specifically designed for a set of GMM (see references). At each iteration, sends information about current prototypes to a server via a socket connection (see references) for info about protocol.

Usage

```
gmmkmsock(models, names, ngroups, rho = new.env(), host = "127.0.0.1")
```

Arguments

models	list of GMM objects.
names	character vector with respective names of the GMM objects.
ngroups	(maximal) number of clusters.
rho	R environment object, used for calls to R functions within C code.
host	IP address of the server for the socket (port 1979).

Value

a set of GMM prototypes, and inferred labels (i.e. associated to the input objects).

Note

gmmkmsock includes a socket client that sends formatted data to a server. Detailed information about this protocol may be found in the source package (inst/doc/old_manual.pdf). Simple standalone client and server are also provided (socket/socketclient.cpp and socketserver.cpp). These can be build by running make in the source folder.

References

P. Bruneau, F. Picarougne, and M. Gelgon. Interactive unsupervised classication and visualization for browsing an image collection. *Pattern Recognition*, 43(2):485–493, 2010

Examples

```
temp1 <- sample(1:200, 150)
temp2 <- list()
for(i in 1:length(temp1)) temp2 <- appendToList(temp2, imgmods[[temp1[i]]])
temp3 <- imgnames[temp1]
# next command may be executed only if a server is running on 127.0.0.1:1979.
# temp4 <- gmmkmsock(temp2, temp3, 5)
```

gmmpen *gmmpen*

Description

list of 10 GMM objects, estimated on subsets of the original 10992-elements pendat data set.

Format

The format is: List of 10 GMM objects

Examples

```
temp <- gmmgen(gmmpen[[1]], 1000)
```

gmmToMppca *gmmToMppca*

Description

uses eigen decompositions to convert a GMM to a MPPCA model.

Usage

```
gmmToMppca(model, alpha = 500)
```

Arguments

model	GMM to be converted.
alpha	GMM are associated to weights, and MPPCA models to population sizes. alpha is the chosen population size for the output MPPCA.

Value

converted MPPCA model.

See Also

mppcaToGmm

Examples

```
temp <- gmmToMppca(gmmpen[[3]])
```

gramschmidt *gramschmidt*

Description

performs Gram-Schmidt orthogonalization on mat.

Usage

```
gramschmidt(mat)
```

Arguments

mat	matrix object to orthogonalize.

Value

orthogonalized matrix.

See Also

mppca newMppca

Examples

```
temp <- gramschmidt(pcapen[[3]]$wmean[[1]])
```

gridGen *gridGen*

Description

generates a matrix valued with a regular grid of 2D coordinates.

Usage

```
gridGen(xlim = c(-10, 10), ylim = c(-10, 10), step = 50)
```

Arguments

xlim	x bounds.
ylim	y bounds.
step	size of the square matrix.

Value

'grid' matrix

Examples

```
temp <- gridGen()
```

---

handdat *handdat*

Description

matrix 300 x 717 of real row-elements. See reference. May be loaded into R with readDataFile. handdat was built using pixmapToVector and filtering variables with zero entropy.

Format

The format is: num [1:300, 1:717] 10 10 10 10 10 10 10 10 10 10 ...

Source

http://yann.lecun.com/exdb/mnist/

References

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) _Gradient-based learning applied to document recognition_, Proceedings of the IEEE, Volume 86, Number 11, Pages 2278-2324.

Examples

```
temp <- reBuild(handdat[123,], handvoid, handnonvoid, handdomains)
```

---

handdomains *handdomains*

Description

original domains of non-void pixels in the handwritten digits collection, to be used along with reBuild.

Format

The format is: List of 2 $: num [1:717] 0.816 0.251 0.278 0.161 0.412 ... $: num [1:717] 1 1 1 1 1 1 1 1 1 1 ...

Examples

```
temp <- reBuild(handdat[123,], handvoid, handnonvoid, handdomains)
```

handlab *handlab*

Description

vector of numeric labels associated to handdat.

Format

The format is: int [1:300] 0 3 2 0 8 1 3 7 3 7 ...

Source

http://yann.lecun.com/exdb/mnist/

References

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998

Examples

```
handlab[1:10]
```

handnonvoid *handnonvoid*

Description

vector of non-void pixel indices.

Format

The format is: int [1:717] 8 9 10 11 12 13 14 15 16 17 ...

Examples

```
temp <- reBuild(handdat[123,], handvoid, handnonvoid, handdomains)
```

handvoid *handvoid*

Description

vector of void pixel indices.

Format

The format is: num [1:67] 1 2 3 4 5 6 7 18 21 24 ...

Examples

```
temp <- reBuild(handdat[123,], handvoid, handnonvoid, handdomains)
```

imglabels *imglabels*

Description

vector of numeric labels, indicating the sub-directory in the Caltech-256 collection associated to respective elements in imgmods.

Format

The format is: num [1:200] 1 1 1 1 1 1 1 1 1 1 ...

Examples

```
imglabels[1:10]
```

imgmods *imgmods*

Description

list of 200 3D GMM, sampled from the 1243 images in the 10 first categories of the Caltech-256 image collection. Built using RGBtoLab and varbayes. See reference for information about this image collection.

Format

The format is: List of 200 GMM

References

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. *http://authors.library.caltech.edu/7694, Technical Report 7694, California Institute of Technology*, 2007

Examples

```
temp <- gmmgen(imgmods[[10]], 1000)
```

---

imgnames *imgnames*

Description

absolute file paths of respective elements in imgmods.

Format

vector of character objects.

Examples

```
imgnames[1:10]
```

---

incremMerge *incremMerge*

Description

updates a reference MPPCA model with an input distribution.

Usage

```
incremMerge(modref, newmod, k = 200, nit = 100, quick = FALSE)
```

Arguments

modref	reference MPPCA to update.
newmod	new MPPCA to incorporate.
k	number of components of the output variational posterior.
nit	number of iterations used in the mmppca call that performs the update.
quick	boolean parameter transmitted to the subMppca routine that shrinks the output variational posterior.

Value

updated variational posterior.

See Also

mppca mmppca

Examples

```
# commented for packaging needs (requires approx. 5s)
#temp <- incremMerge(pcapen[[1]], pcapen[[2]], quick=T)
```

---

irisdata *irisdata*

Description

matrix 150 x 4 of row-elements, extracted from iris standard data.frame (4 first variables). See reference.

Format

The format is: num [1:150, 1:4] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"

References

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 Part II:179–188, 1936

Examples

```
displayScatter(irisdata)
```

---

irislabels *irislabels*

Description

vector of numeric labels associated to irisdata.

Format

The format is: num [1:150] 1 1 1 1 1 1 1 1 1 1 ...

Examples

```
displayScatter(data=irisdata, labels=irislabels)
```

isNonVoid *isNonVoid*

Description

checks if loadings contains only void columns.

Usage

```
isNonVoid(loadings)
```

Arguments

loadings	matrix from which we check the columns.

Value

TRUE if at least 1 column is not void.

See Also

mppca newMppca

Examples

```
isNonVoid(pcapen[[1]]$wmean[[2]])
#[1] TRUE
```

jsmc *jsmc*

Description

computes Monte Carlo estimate of Jensen-Shannon (JS) divergence between GMM.

Usage

```
jsmc(mod1, mod2, nsamp = 5000)
```

Arguments

mod1	GMM parameter to JS(mod1 ‖ mod2).
mod2	GMM parameter to JS(mod1 ‖ mod2).
nsamp	number of samples used to build estimate.

Value

JS divergence value.

See Also

klmc gaussianKL

Examples

```
temp <- jsmc(gmmpen[[1]], gmmpen[[2]])
```

`jsut` *jsut*

Description

compute Unscented Transform approximation to Jensen-Shannon (JS) divergence between GMM.

Usage

```
jsut(mod1, mod2)
```

Arguments

`mod1`	GMM parameter to JS(mod1 ‖ mod2).
`mod2`	GMM parameter to JS(mod1 ‖ mod2).

Value

JS divergence value.

References

J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. *ICCV Proceedings*, 1:487–493, 2003

See Also

klut jsmc

Examples

```
temp <- jsut(gmmpen[[1]], gmmpen[[2]])
```

klmc *klmc*

Description

computes Monte Carlo estimate of KL divergence between GMM.

Usage

```
klmc(mod1, mod2, nsamp = 5000)
```

Arguments

mod1	GMM parameter to KL(mod1 || mod2).
mod2	GMM parameter to KL(mod1 || mod2).
nsamp	number of samples used to build estimate.

Value

KL value.

See Also

jsmc gaussianKL

Examples

```
temp <- klmc(gmmpen[[1]], gmmpen[[2]])
```

klut *klut*

Description

compute Unscented Transform approximation to KL divergence between GMM.

Usage

```
klut(mod1, mod2)
```

Arguments

mod1	GMM parameter to KL(mod1 || mod2).
mod2	GMM parameter to KL(mod1 || mod2).

Value

KL value.

References

J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. *ICCV Proceedings*, 1:487–493, 2003

See Also

klmc

Examples

```
temp <- klut(gmmpen[[1]], gmmpen[[2]])
```

`l2norm` *l2norm*

Description

computes Euclidian norm of vec.

Usage

```
l2norm(vec)
```

Arguments

`vec`	numeric vector.

Value

norm value.

Examples

```
temp <- l2norm(gmmpen[[2]]$mean[[1]])
```

mergeClassif *mergeClassif*

Description

performs task analogous to mixKnn (i.e. leave-one-out classification), but uses synthetic representatives to infer labels, instead of k-NN. Each representative is obtained by concatenating all GMM (i.e. elements) of a specific label value, and applying vbcomp on this redundant mixture.

Usage

```
mergeClassif(data, labels, KLparam = 500, rho = new.env())
```

Arguments

data	list of GMM.
labels	vector of numeric labels associated to data.
KLparam	number of samples for jsmc.
rho	R environment object. Used to issue R commands within the C routine.

Value

classification error ratio in [0,1].

See Also

mixKnn vbcomp

Examples

```
temp1 <- sample(1:200, 150)
temp2 <- list()
for(i in temp1) temp2 <- appendToList(temp2, imgmods[[i]])
temp3 <- imglabels[temp1]
# de-activated because this process is very long...
#temp4 <- mergeClassif(temp2, temp3)
```

mixKnn *mixKnn*

Description

performs k-nearest neighbors over a collection of GMM. It uses jsmc to compute distances. Each elements in data is classified against all the others, and inferred class is compared to the true one (leave-one-out).

Usage

```
mixKnn(data, labels, n = 2, KLparam = 500)
```

Arguments

data	list of GMM.
labels	vector of numeric labels associated to data.
n	k of the algorithm.
KLparam	number of samples for jsmc.

Value

classification error ratio in [0,1].

See Also

mergeClassif constrClassif sampleClassif

Examples

```
temp1 <- sample(1:200, 150)
temp2 <- list()
for(i in temp1) temp2 <- appendToList(temp2, imgmods[[i]])
temp3 <- imglabels[temp1]
# de-activated because this process is very long...
#temp4 <- mixKnn(temp2, temp3)
```

mmppca *mmppca*

Description

estimates the variational posterior distribution of a MPPCA that aggregates a collection of input MPPCA models. A lower bound is calculated and monitored at each iteration. This posterior can be used for various purposes (e.g. MC proposal distribution). It can be transformed using mppcaToGmm and subMppca, outputing a GMM. The maximal rank of output factor matrices is determined by the inputs.

Usage

```
mmppca(mods, ncomp, thres = 0.1, maxit = NULL)
```

Arguments

mods	input MPPCA that concatenates the set of components to aggregate.
ncomp	number of components in the posterior.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.

Value

estimated posterior MPPCA with ncomp components.

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

newMppca mppca subMppca

Examples

```
temp <- newMppca()
for(i in 1:3) temp <- appendToMppca(temp, pcapen[[i]])
temp2 <- mmppca(temp, 50, maxit=30)
```

mppca *mppca*

Description

estimates the variational posterior distribution of a MPPCA on a data set. A lower bound is calculated and monitored at each iteration. This posterior can be used for various purposes (e.g. MC proposal distribution). It can be transformed using mppcaToGmm and subMppca, outputing a GMM.

Usage

```
mppca(data, ncomp, thres = 0.1, maxit = NULL, qmax = NULL)
```

Arguments

data	matrix of row-elements.
ncomp	number of components in the posterior.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.
qmax	maximal rank of the posterior factor matrices. If NULL, is set to d-1.

Value

estimated posterior MPPCA with ncomp components.

References

M. J. Beal. Variational algorithms for approximate inference. *PhD Thesis, University of London*, 2003

See Also

newMppca mppcaToGmm subMppca

Examples

```
# for packaging needs, a low amount of initial components (ie 10) was used.
# A larger amount may be used for better results.
temp <- mppca(pendat, 10, maxit=20, qmax=8)
```

mppcaToGmm *mppcaToGmm*

Description

converts a MPPCA model to a GMM.

Usage

```
mppcaToGmm(model, notau = FALSE)
```

Arguments

model	MPPCA model to be converted.
notau	if TRUE, covariances are built with Lambda.Lambda^T without adding tau.

Value

GMM after conversion.

References

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society - B Series*, 61(3):611–622, 1999

See Also

mppca varbayes

Examples

```
temp <- mppcaToGmm(pcapen[[1]])
```

multinomial *multinomial*

Description

samples from a k-multinomial.

Usage

```
multinomial(weights, k)
```

Arguments

weights	numeric vector with the weights of the multinomial. Sum to 1.
k	size of the weight vector.

Value

an integer value in [1,k], coded as a 1-of-k variable (see reference).

References

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006

Examples

```
weights <- c(0.3, 0.5, 0.2)
multinomial(weights, 3)
#[1] 0 1 0
```

---

mvndensity *mvndensity*

Description

get densities of a set of elements w.r.t a multivariate normal.

Usage

```
mvndensity(mean, cov, data)
```

Arguments

mean	numeric vector, mean of the multivariate normal.
cov	covariance matrix of the multivariate normal.
data	matrix of row-elements.

Value

numeric vector containing densities.

See Also

mvngen

Examples

```
temp <- mvngen(c(0, 0), diag(2), 5)
mvndensity(c(0,0), diag(2), temp)
#[1] 0.137188286 0.032318242 0.005181099 0.047312602 0.033178600
```

mvngen *mvngen*

Description

sample nitem elements from N(mean, cov).

Usage

```
mvngen(mean, cov, nitem)
```

Arguments

mean	numeric vector.
cov	covariance matrix.
nitem	number of items to generate.

Value

nitem x d matrix with elements as rows (further denoted as a matrix of row-elements).

Examples

```
mvngen(c(0, 0), diag(2), 5)
#       [,1]        [,2]
#[1,] -0.09898211  1.4516438
#[2,]  0.20814926 -0.1233861
#[3,]  0.18410071  0.5995621
#[4,]  0.65994562  0.8328315
#[5,]  2.33098055 -0.5943117
```

mymvn2plot *mymvn2plot*

Description

displays mvn envelopes. For internal usage in graphical functions.

Usage

```
mymvn2plot(w, mu, sigma, k = 15, alone = FALSE, col = NA, alphacol = 0.8, alphanocol = 0.5, lty = "solid")
```

Arguments

w	weight of the component.
mu	mean of the component.
sigma	covariance matrix of the component.
k	resolution used for drawaing the elliptic envelope.
alone	if TRUE, the component is to be plotted alone in its own window.
col	optional background color for the component.
alphacol	alpha coefficient for a component with a color.
alphanocol	alpha coefficient for a component with no color.
lty	line type for the ellipsis.

---

mySmoothScatter *mySmoothScatter*

Description

Personalized version of smoothScatter. For internal usage in graphical functions.

Usage

```
mySmoothScatter(data, model, xlim, ylim)
```

Arguments

data	matrix of row-elements to plot.
model	Optional Gaussian components to plot.
xlim	optional bound for plotting.
ylim	optional bound for plotting.

---

newGmm *newGmm*

Description

creates an empty GMM data structure.

Usage

```
newGmm()
```

Value

list object with the following members:

w	numeric vector containing the component weights of the mixture model.
mean	list with respective means (numeric vectors) as elements.
cov	list with respective covariance matrices as elements.
a	constraints between components, encoded in a numeric vector. One value per component. 2 components associated to the same value are said to be from the same origin. Used in vbconstr.

See Also

varbayes vbconstr

Examples

```
temp <- newGmm()
```

---

newMppca *newMppca*

Description

creates an empty posterior MPPCA data structure.

Usage

```
newMppca()
```

Value

list object with the following members:

alpha	numeric vector for bayesian alpha parameter.
numoment	list of numeric vectors, containing E[nu_(kj)] parameters.
nub	list of numeric vectors, containing b_(kj) parameters for nu.
taumoment	numeric vector for tau parameter. NB: all set identically and statically to 1, as in [Bruneau 2011] a single static tau parameter is used.
taua	numeric vector for a_k parameters for tau.
taub	numeric vector for b_k parameters for tau.
wmean	list of matrices containing E[Lambda_k] parameters.
wsigma	list of matrices containing Cov(Lambda_k^(i.)).
xsigma	list of matrices containing Cov(x_k).
mumean	list of numeric vectors, containing means of the MPPCA model.
musigma	list of matrices with covariances for the mean estimates.
mustar	list of numeric vectors, containing prior means of the MPPCA model, used for initialisation.

References

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

mppca mmppca

Examples

```
temp <- newMppca()
```

`normalizeVariable` *normalizeVariable*

Description

normalizes a variable (numeric vector) in [0,1].

Usage

```
normalizeVariable(v)
```

Arguments

v	a numeric vector.

Value

normalized numeric vector.

Examples

```
temp <- normalizeVariable(irisdata[,1])
```

normMppca *normMppca*

Description

adjusts a MPPCA model to ensure that all factor matrices have same rank (q).

Usage

```
normMppca(mppca1)
```

Arguments

mppca1	MPPCA model to be adjusted.

Value

adjusted MPPCA.

See Also

newMppca mppca

Examples

```
temp <- newMppca()
for(i in 1:5) temp <- appendToMppca(temp, pcapen[[i]])
temp <- normMppca(temp)
```

pca *pca*

Description

transforms a data set, and returns coordinates in the principal basis.

Usage

```
pca(dat, ncomp = NULL)
```

Arguments

dat	matrix of row-elements.
ncomp	number of retained variables in the output result. If NULL, all transformed variables are returned.

Value

matrix of transformed row-elements.

References

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society - B Series*, 61(3):611–622, 1999

P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012

See Also

mppca

Examples

```
temp <- pca(irisdata, 3)
```

---

pcapen *pcapen*

Description

list of 10 MPPCA posterior objects, estimated on subsets of the original 10992-elements pendat data set.

Format

The format is: List of 10 posterior MPPCA objects

Examples

```
temp <- mppcaToGmm(pcapen[[1]])
```

pendat *pendat*

Description

matrix 2000 x 16 of real row-elements.

Format

The format is: num [1:2000, 1:16] -4.6 -1.2 -2.4 8.4 0.6 3.8 -10 8.8 -10 4.4 ...

Source

http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

References

F. Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. *Technical report, Institute of Graduate Studies in Science and Engineering*, 1996

Examples

```
displayScatter(pendat)
```

penlab *penlab*

Description

vector of numeric labels associated to pendat.

Format

The format is: int [1:2000] 5 3 8 6 0 9 1 8 1 9 ...

Source

http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

References

F. Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. *Technical report, Institute of Graduate Studies in Science and Engineering*, 1996

Examples

```
displayScatter(data=pendat, labels=penlab)
```

pixmapToVector *pixmapToVector*

Description

converts a pixmapGrey object to a numeric vector. The pixel matrix is casted to a vector by appending successive columns.

Usage

```
pixmapToVector(p)
```

Arguments

p	pixmapGrey object.

Value

numeric vector containing pixel intensities.

See Also

pixmapGrey reBuild readPixmapFile

Examples

```
# use with path to actual train-... file
#temp <- readPixmapFile("data/train-images-idx3-ubyte")
#temp2 <- pixmapToVector(temp[[3]])
```

plotGmm *plotGmm*

Description

3D density plot of a 2D GMM.

Usage

```
plotGmm(mod, steps=200)
```

Arguments

mod	GMM object to plot
steps	specifies the hoizontal and vertical amount of vertices used to build the wireframe plot.

Value

a new plotting window with the 3D density plot.

See Also

displayScatter

Examples

```
# a larger number of steps (eg 200) should be used for a visually effective 3D plot.
plotGmm(randomGmm(), steps=20)
```

---

randomGmm *randomGmm*

Description

sample randomly a GMM. Number of components is sampled from a Poisson law, means uniformly from [-domain, domain], and covariance matrices using covgen function.

Usage

```
randomGmm(domain = 10)
```

Arguments

domain	determines the domain from which means are sampled.

Value

randomly sampled GMM.

See Also

covgen newGmm

Examples

```
temp <- randomGmm()
```

Rdct *Rdct*

Description

performs DCT on a real vector.

Usage

```
Rdct(vect)
```

Arguments

vect	vector of real values.

Value

vector of DCT transformed values.

See Also

Rdct2D

Examples

```
temp <- Rdct(irisdata[,1])
```

Rdct2D *Rdct2D*

Description

performs 2D DCT on a real matrix.

Usage

```
Rdct2D(mat)
```

Arguments

mat	matrix of real values.

Value

matrix of DCT transformed values.

See Also

Rdct RinvDct2D

Examples

```
temp <- Rdct2D(irisdata)
```

`rDirichlet` *rDirichlet*

Description

samples from the Dirichlet distribution.

Usage

```
rDirichlet(K, alpha = 0.1)
```

Arguments

`K`	order of the sample.
`alpha`	alpha parameter of the distribution (i.e. alpha repeated K times).

Value

numeric vector, which values are in [0,1] and sum to 1.

See Also

dDirichlet

Examples

```
temp <- rDirichlet(4)
```

readLabelFile *readLabelFile*

Description

reads the vector of numeric labels contained in a binary file. Labels are associated to handwritten digits, thus in [0-9].

Usage

```
readLabelFile(name)
```

Arguments

name	path to the file.

Value

vector of digit labels.

References

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998

http://yann.lecun.com/exdb/mnist/, repository of the labels file.

See Also

readPixmapFile

Examples

```
# use with path to actual train-...
# temp <- readLabelFile("data/train-labels-idx1-ubyte")
```

readPixmapFile *readPixmapFile*

Description

extracts a list of pixmap objects from the handwritten digits file format provided in references.

Usage

```
readPixmapFile(name)
```

Arguments

name	path to the file.

Value

a list of pixmapGrey objects.

References

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998

http://yann.lecun.com/exdb/mnist/, repository of the labels file.

See Also

pixmapGrey

Examples

```
# use with path to actual train-... file
#temp <- readPixmapFile("data/train-images-idx3-ubyte")
```

---

reBuild *reBuild*

Description

re-build a pixmapGrey object from a vector of pixel intensities. As some pixels may be irrelevant over a collection of images (e.g. pixel always white in handwritten digits), some variables may have been filtered or transformed before performing some machine learning process. These transforms are indicated as parameters, and give clues to recover objects in the original image space. NB: assumes that v is scaled in [-10,10]. Additional transformations may thus be performed as appropriate before using this function.

Usage

```
reBuild(v, voids, nonvoids, domains, placeholder = 1)
```

Arguments

v	vector to be converted to a pixmapGrey object.
voids	vector of position indices in the original signal (i.e. 2D matrix with its columns casted in a vector) that did not carry any information. Replaced by a placeholder in recovered image.
nonvoids	vector of positions to which v should be associated in the recovered image.

domains	original data domains of pixel intensities prior to being transformed to v's domain. Permit appropriate reconstruction in the domain of pixel intensities used by pixmap (i.e. subset of [0,1]). Formatted similarly to what is required in setDomain.
placeholder	placeholder value for pixel positions present in voids.

Value

pixmapGrey reconstructed object.

See Also

pixmapGrey pixmapToVector

Examples

```
temp <- reBuild(handdat[123,], handvoid, handnonvoid, handdomains)
```

RGBtoLab *RGBtoLab*

Description

transform a .ppm file into a matrix of (L,a,b) pixel intensities (1 row-element per pixel).

Usage

```
RGBtoLab(filename, filterWhite = FALSE, addCoords = TRUE)
```

Arguments

filename	path to a .ppm file. Alternatively, if needed, R file path manipulating routines are documented in document r-lang.pdf, section 7.1)
filterWhite	if TRUE, filter white points from result to return.
addCoords	if TRUE, append 2 normalized (x,y) coordinates for each pixel.

Value

matrix of pixel row-elements.

Note

In order to save space, images associated to names in imgnames were not provided in this bundle. Caltech-256 should be retrieved first, converted to .ppm (e.g. with imageMagick), and then values in imgnames associated to relevants file paths, before using RGBtoLab.

Examples

```
# image collections are large, thus not provided. The following commented example relates to a member of this image c
#temp <- RGBtoLab(imgnames[[2]], filterWhite=TRUE)
```

RinvDct2D *RinvDct2D*

Description

performs inverse 2D DCT on a real matrix.

Usage

```
RinvDct2D(mat)
```

Arguments

mat	matrix of real values.

Value

matrix of inverse DCT transformed values.

See Also

Rdct Rdct2D

Examples

```
temp <- RinvDct2D(Rdct2D(irisdata))
```

sampleClassif *sampleClassif*

Description

performs task analogous to mixKnn (i.e. leave-one-out classification), but uses synthetic representatives to infer labels, instead of k-NN. Each representative is obtained by concatenating all GMM (i.e. elements) of a specific label value, resampling from this redundant mixture, and applying varbayes on this sample.

Usage

```
sampleClassif(data, labels, KLparam = 500, rho = new.env())
```

Arguments

data	list of GMM.
labels	vector of numeric labels associated to data.
KLparam	number of samples for jsmc.
rho	R environment object. Used to issue R commands within the C routine.

Value

classification error ratio in [0,1].

See Also

mixKnn

Examples

```
temp1 <- sample(1:200, 150)
temp2 <- list()
for(i in temp1) temp2 <- appendToList(temp2, imgmods[[i]])
temp3 <- imglabels[temp1]
# de-activated because this process is very long...
#temp4 <- sampleClassif(temp2, temp3)
```

---

semispheregen *semispheregen*

Description

sample data points along a semi-sphere.

Usage

```
semispheregen(npts = 200, radius = 10, noise = 1)
```

Arguments

npts	number of elements to be sampled.
radius	radius of the sphere.
noise	additive gaussian white noise to the sampled points.

Value

matrix of row-elements with the sampled elements.

Examples

```
temp <- semispheregen()
```

setDomain *setDomain*

Description

performs linear rescaling of given data.

Usage

```
setDomain(dat, span = 10, oldspan = NULL)
```

Arguments

dat	data to rescale. matrix object, with elements as rows, and variables as columns (i.e. variables are rescaled).
span	new domain to which dat is rescaled. If type is numeric and length = 1: [-span, span] is used for all variables. If type is numeric and length = 2: [span[1], span[2]] is used for all variables. If a list object: [span[[1]]_i, span[[2]]_i] is used for each variable i.
oldspan	if NULL, old domains are computed from dat inspection. Otherwise, is structured as span and replaces inspected values for rescaling.

Value

scaled data matrix.

Examples

```
temp <- setDomain(irisdata, span=15)
```

sort_index *sort_index*

Description

returns indexes associated to the sorted values of the parameter vector.

Usage

```
sort_index(vec, order = 0)
```

Arguments

vec	vector to be sorted.
order	if 0, ascending order, if 1, descending order.

Value

indexes associated to the sorted input vector.

Examples

```
temp <- rnorm(10)
temp2 <- sort_index(temp)
```

`spiralgen` *spiralgen*

Description

generates data elements along a spiral with additional noise.

Usage

```
spiralgen(radius = 10, n = 1000, laps = 2, noise = 1)
```

Arguments

`radius`	determines the radius of a spiral revolution.
`n`	number of elements to generate.
`laps`	number of revolutions of the spiral.
`noise`	determines the width of the spiral stroke.

Value

matrix of sampled row-elements.

See Also

datagen circlegen

Examples

```
temp <- spiralgen()
```

subGmm *subGmm*

Description

select a subset of components and dimensions from an input GMM.

Usage

```
subGmm(model, dims = c(1, 2), inds = NULL)
```

Arguments

model	GMM from which to extract subsets.
dims	numeric vector of the extracted dimensions.
inds	numeric vector of selected components indices. If NULL, all components are selected.

Value

subset of input GMM.

See Also

newGmm

Examples

```
temp <- subGmm(gmmpen[[1]], inds=1:3)
```

subMppca *subMppca*

Description

removes unused components and factor columns from model.

Usage

```
subMppca(model, prune = FALSE, thres = 2.001, quick = FALSE, noxmean = TRUE)
```

Arguments

model	MPPCA model to be shrinked.
prune	if TRUE, void factor columns are removed.
thres	threshold for component selection. A components is selected iif alpha > thres.
quick	influences method for void factor columns detection. If FALSE, a KL-based criterion is employed (more accurate). If TRUE, column norms are used (useful for very high dimensional data sets).
noxmean	should always be set to TRUE.

Value

shrinked MPPCA model.

See Also

mppca newMppca

Examples

```
# use a subsample of pendat, for runtime (packaging) needs.
temp <- mppca(pendat[sample(1:2000,150),], 15, qmax=8, maxit=20)
temp2 <- subMppca(temp, prune=TRUE, quick=TRUE)
```

---

subVarbayes *subVarbayes*

Description

filters a variational posterior GMM, keeping only components with sufficient support.

Usage

```
subVarbayes(model, thres = 2.001)
```

Arguments

model	variational posterior GMM.
thres	minimal support for component selection.

Value

filtered variational posterior GMM.

See Also

varbayes extractSimpleModel

Examples

```
temp <- varbayes(irisdata, 20)
temp2 <- subVarbayes(temp)
```

---

varbayes *varbayes*

Description

estimates the variational posterior distribution of a GMM on data using the variational EM algorithm (see references). A lower bound is calculated and monitored at each iteration. This posterior can be used for various purposes (e.g. MC proposal distribution). It can be transformed using extractSimpleModel, outputing a GMM.

Usage

```
varbayes(data, ncomp, thres = 0.1, maxit = NULL)
```

Arguments

data	matrix of row-elements.
ncomp	number of components in the posterior.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.

Value

estimated posterior with ncomp components. Structured in a list object as follows:

alpha	hyperparameters influencing the active components in the posterior.
beta	hyperparameters regarding shaping of the Normal-Wishart posteriors.
nu	hyperparameters regarding shaping of the Normal-Wishart posteriors.
mean	hyperparameters regarding shaping of the Normal-Wishart posteriors.
wish	hyperparameters regarding shaping of the Normal-Wishart posteriors.

References

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006

See Also

classicEM extractSimpleModel

Examples

```
temp <- varbayes(irisdata, 20)
```

vbcomp *vbcomp*

Description

estimates the variational posterior distribution of a GMM that aggregates a collection of GMM. A lower bound is calculated and monitored at each iteration. This posterior can be used for various purposes (e.g. MC proposal distribution). It can be transformed using extractSimpleModel, outputing a GMM.

Usage

```
vbcomp(models, ncomp, thres = 0.1, maxit = NULL)
```

Arguments

models	GMM made with the weighted sum of the collection of GMM to aggregate.
ncomp	number of components in the posterior.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.

Value

estimated posterior with ncomp components.

References

P. Bruneau, M. Gelgon, and F. Picarougne. Parsimonious reduction of gaussian mixture models with a variational-bayes approach. *Pattern Recognition*, 43:850–858, 2010

See Also

varbayes extractSimpleModel

Examples

```
temp1 <- newGmm()
for(i in 1:10) temp1 <- appendToGmm(temp1, gmmpen[[i]])
temp2 <- vbcomp(temp1, 50)
```

vbconstr *vbconstr*

Description

estimates the variational posterior distribution of a GMM that aggregates a constrained collection of GMM. A lower bound is calculated and monitored at each iteration. This posterior can be used for various purposes (e.g. MC proposal distribution). It can be transformed using extractSimpleModel, outputing a GMM.

Usage

```
vbconstr(models, ncomp, thres = 0.1, maxit = NULL)
```

Arguments

models	GMM made with the weighted sum of the collection of GMM to aggregate. a is used to model constraints between components in this GMM.
ncomp	number of components in the posterior.
thres	threshold for lower bound variations between 2 iterations. Convergence is decided if this variation is below thres.
maxit	if NULL, the stopping criterion is related to thres. If not NULL, maxit iterations are performed.

Value

estimated posterior with ncomp components.

References

P. Bruneau, M. Gelgon, and F. Picarougne. Parsimonious reduction of gaussian mixture models with a variational-bayes approach. *Pattern Recognition*, 43:850–858, 2010

See Also

vbcomp extractSimpleModel

Examples

```
temp1 <- newGmm()
for(i in 1:10) temp1 <- appendToGmm(temp1, gmmpen[[i]])
temp2 <- vbconstr(temp1, 50)
```

vbpen *vbpen*

Description

list of 10 variational posterior GMM objects, estimated on subsets of the original 10992-elements pendat data set.

Format

The format is: List of 10 variational GMM.

Examples

```
temp <- extractSimpleModel(vbpen[[2]])
```

ZtoLabels *ZtoLabels*

Description

converts a responsibility matrix (Z in references) to a vector of numeric labels.

Usage

```
ZtoLabels(resp)
```

Arguments

resp	responsibility matrix to convert.

Value

labels vector.

References

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006

See Also

getResp getVarbayesResp

Examples

```
temp <- getResp(pendat, pcapen[[2]])
temp2 <- ZtoLabels(temp)
```

Bibliography

[1] F. Alimoglu. Combining multiple classifiers for pen-based handwritten digit recognition. *Technical report, Institute of Graduate Studies in Science and Engineering*, 1996.

[2] M. J. Beal. Variational algorithms for approximate inference. *PhD Thesis, University of London*, 2003.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[4] P. Bruneau, M. Gelgon, and F. Picarougne. Parsimonious reduction of gaussian mixture models with a variational-bayes approach. *Pattern Recognition*, 43:850–858, 2010.

[5] P. Bruneau, M. Gelgon, and F. Picarougne. A low-cost variational-bayes technique for merging mixtures of probabilistic principal component analyzers. *Information Fusion*, 2012.

[6] P. Bruneau, F. Picarougne, and M. Gelgon. Interactive unsupervised classication and visualization for browsing an image collection. *Pattern Recognition*, 43(2):485–493, 2010.

[7] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 Part II:179–188, 1936.

[8] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *J. Am. Stat. Assoc.*, 78:553–569, 1983.

[9] J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. *ICCV Proceedings*, 1:487–493, 2003.

[10] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. *http://authors.library.caltech.edu/7694, Technical Report 7694, California Institute of Technology*, 2007.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] F. Picarougne, H. Azzag, G. Venturini, and C. Guinot. A new approach of data clustering using a flock of agents. *Evolutionary Computation*, 78(3):345–367, 2007.

[13] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 61:461–464, 1978.

[14] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society - B Series*, 61(3):611–622, 1999.